IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application		)	PATENT APPLICATION
Inventors:	Cirne, et al.	)	
Application No.:	10/700,338	)	Art Unit: 2192
Filed:	November 3, 2003	)	Examiner: Wei, Zheng
Title:	SIMPLE METHOD OPTIMIZATION	)	Customer No. 28554

AMENDMENT TO APPEAL BRIEF

Mail Stop Appeal Brief – Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This Amendment is in response to the Notification of Non-Compliant Appeal Brief, dated March 20, 2008.

This Amendment is submitted in accordance with 37 C.F.R. §41.37, following the Notice of Appeal filed by Appellants on January 9, 2008 and the Appeal Brief filed by Appellants on March 10, 2008. The fee set forth in 1.17(c) was submitted on March 10, 2008.

AMENDMENT TO SECTION V - SUMMARY OF CLAIMED SUBJECT MATTER *(37 C.F.R. §41.37(c)(v))*

The technology recited in claims 1-3, 5-15, 17-24, 26-35, 37-42, and 44-52 generally relate to "technology for monitoring applications" (Specification p. 2, lines 16-17). As discussed in the Background of the Invention section, prior monitoring techniques, such as the implementation of a performance analysis tool, provide an indication of an application's performance. These indications may, for example, "provide timing data on how long each method (or procedure or other process) is being executed, report how many times each method is executed and/or identify the function call architecture" (Specification p. 1, lines 22-25). However, many times these tools provide too much data, which problematically creates for the user the "difficult task of analyzing the multitude of data to determine which data is relevant and which data is not relevant (e.g. redundant or otherwise not useful)" (Specification p. 2, lines 3-4). In many instances, prior monitoring techniques would "instrument a large number of methods in the software in order to be able to analyze the performance of each method. However, modifying a large number of methods... [adds] an enormous amount of code to the software and may impact performance of the underlying software" (Specification p. 2, lines 11-13).

The technology recited in the claims for the present application resolves these issues by only modifying certain methods for purposes such as monitoring or tracing, making the process of monitoring or tracing applications much more efficient and useful. Specifically, the technology of the present application modifies methods that are determined to be complex. Methods that are determined to be simple are not modified. As stated in the Specification:

> In one embodiment, a method is complex if it meets three criteria: (1) the method has an access level of public or package; (2) the method is non-synthetic and (3) the method calls at least one other method. Methods that do not satisfy all three criteria are classified as simple methods. In other embodiments, a method can be classified as complex if it satisfies two of the above criteria, or other similar criteria (p. 12, lines 17-23).

Software typically contains methods that call other methods. For example, in Figure 4 of the Drawings, method M1 calls methods M2 and M3, and method M3 calls methods M4 and M5, both of which do not call another method (also see Specification p. 11, line 23 – p. 12, line 11). Monitoring M1 would result in the monitoring of all of the methods that it calls. It would be inefficient and redundant to also monitor the methods that it calls separately. Therefore, it is useful to monitor only methods which call at least one other methods.

The Specification describes a method's access level as follows:

> Java provides for four levels of access control for methods: public, private, protected, and package. Private is the most restrictive access level. A private method can only be accessed by methods in the same class. A protected method can be accessed by other methods in the same class, sub classes and classes in the same package. A public method can be accessed by any class of any parentage in any package. The package access level is the default and allows a method to be accessed by classes in the same package, regardless of their parentage" (p. 12, line 24 – p. 13, line 5).

In the example code found on page 13, line 13 to page 14, line 9 of the Specification, the access level can be determined by referencing the access level indicated prior to each method name (e.g. public methodOne()).

A method is determined to be synthetic if it is "a method that does not appear in the source code. For example, during compilation, the compiler may add one or more methods. Typically, the compiler explicitly makes it easy to see that methods are or are not synthetic. Java compilers flag these methods with the 'Synthetic' attribute" (Specification p. 13, lines 6-11).

As shown in Figure 5 of the Drawings, "various methods are accessed and determined to be either complex or simple… If the method is complex, it is modified… If the method is determined to be simple, it is not modified for the purposes that the complex methods are modified" (Specification p. 14, lines 19-23). Figure 6 of the Drawings shows one way a method that is determined to be complex can be modified. A method is modified by accessing the beginning of the byte code for the method (step 302), adjusting the byte code indices to prepare for the addition of code (step 304), adding new start byte code (step 306), accessing the end of the byte code (step 308), adding new exit

byte code (step 310), adjusting the exception table due to the adjustment of the byte code indices (step 312), and adding a new exception table entry in the exception table (step 314) (also see Specification p. 15, line 14 – p. 16, line 21). Such modification to the code for a method can be performed for any purpose, such as monitoring or tracing the method.

The method recited in independent claim 1 relates to a "process for monitoring." Figure 3 of the Drawings describes [[a]] one process for monitoring using a modified method, which includes receiving existing code (step 260), receiving new function(s) such as "new classes and methods that allow for monitoring the application" (step 262), modifying existing code (as described in Figure 6, for example) (step 264), adding "all or part of the new functionality (e.g. the new classes/methods)… [to] the existing code" (step 266), storing the modified code (step 268), and running the modified code for monitoring (step 270) (Specification p. 10, lines 10-21). The Specification describes that the process for monitoring may be implemented, for example, through "software that is stored on one or more processor readable storage devices [that] is used to program one or more processors (p. 10, lines 1-2).

Claim 1 further recites, "accessing a method." One example of this feature is shown in step 280 of Figure 5 of the Drawings. The "method is accessed from the set of methods that are to be considered" for monitoring (Specification p. 14, lines 24-25).

Claim 1 further recites "determining whether to modify said method, said step of determining whether to modify said method includes determining whether said method calls another method." One example of this feature is shown in Figure 5 of the Drawings in the decision block of step 286. "If the method under consideration does not call any other methods, then that method is not modified for the purposes that the complex methods are modified (step 290). If the method under consideration does call one or more methods, then the method under consideration is modified" (Specification p. 15, lines 6-8).

Claim 1 further recites "modifying said method for a particular purpose if said method calls another method." As stated above and shown in Figure 5 of the Drawings, the method is modified if the method calls other methods (step 288). The method can be modified for a particular purpose, such as "monitoring [an] application" (Specification p. 10, line 13), for example. The monitoring for a particular purpose may include the addition of "functionality in order to reduce overhead and reduce the amount of

data generated, without losing important data" (Specification p. 10, lines 23-24).

The method recited in claim 13 also relates to a "process for monitoring." The process for monitoring is carried out by "determining which methods of a set of methods call one or more other methods." This step of determining, as mentioned above, is described and shown in Figure 5, step 286 and Specification p. 10, lines 10-21.

Claim 13 further recites "using a first tracing mechanism for said methods determined to call one or more other methods without using said first tracing mechanism for methods not determined to call one or more other methods." Figure 5 of the Drawings shows that if the method is determined to call other methods (step 286), the method will be modified (step 288) "to add the tracer" (Specification p. 15, line 9). "If the method under consideration does not call any other methods, then that method is not modified for the purposes that the complex methods are modified," such as to add the tracer (Specification p. 15, lines 6-9). The tracing mechanism is used when the modified code is run (step 270 of Figure 3 of the Drawings).

The technology recited in claim 22 includes "one or more processor readable storage devices having processor readable code embodied on said processor readable storage devices, said processor readable code for programming one or more processors to perform a process." The Specification describes that the process for monitoring may be implemented through "software that is stored on one or more processor readable storage devices [that] is used to program one or more processors (p. 10, lines 1-2).

Claim 22 further recites "determining which methods of a set of methods to modify, said step of determining includes determining whether said methods call one or more other methods." This step of determining, as mentioned above, is described and shown at least in Figure 5, step 286 and Specification p. 10, lines 10-21.

Claim 22 further recites "modifying for a particular purpose those methods that are determined to call one or more other methods." This step of modifying, as mentioned above for claim 1, is described and shown at least in Figure 5, step 288 of the Drawings and on page 10 of the Specification.

The technology recited in claim 33, like claim 22, also includes "one or more processor readable storage devices." Claim 33 further recites "determining whether to trace a method, said step of determining includes determining whether to said method calls another method." This feature, as

discussed above for claim 1, is shown in Figure 5 of the Drawings in the decision block of step 286 and Specification p. 10, lines 10-21.

Claim 33 further recites "tracing said method for a particular purpose if said method calls another method. "If the method under consideration does call one or more other methods, then the method under consideration is modified to add a tracer" (Specification p. 15, lines 8-9). When the modified code is run, the method is traced (step 270 of Figure 3 of the Drawings).

Claim 39 recites means plus function language for an "apparatus capable of monitoring." The claim recites "means for determining whether a method calls another method." The Specification discloses that the "means for determining" can be "implemented in software that is stored on one or more processor readable storage devices… to program one or more processors" (p. 9, line 27 – p. 10, line 2). The "means for determining" can further be incorporated in "computing devices [which] will include [the] one or more processor in communication with one or more processor readable storage devices" (p. 9, lines 18-20). The determining whether a method calls another method is discussed and shown in Figure 5 of the Drawings in the decision block of step 286. "If the method under consideration does not call any other methods, then that method is not modified for the purposes that the complex methods are modified (step 290). If the method under consideration does call one or more methods, then the method under consideration is modified" (Specification p. 15, lines 6-8).

Claim 39 further recites "means for tracing said method for a particular purpose only if said method calls another method." The "means for tracing" can be "implemented in software that is stored on one or more processor readable storage devices," much like the "means for determining" (p. 9, line 27 – p. 10, line 2). Tracing the method is discussed and shown in step 288 of Figure 5 of the Drawings. Figure 5 of the Drawings shows that if the method is determined to call other methods (step 286), the method will be modified (step 288) "to add the tracer" (Specification p. 15, line 9). "If the method under consideration does not call any other methods, then that method is not modified for the purposes that the complex methods are modified," such as to add the tracer (Specification p. 15, lines 6-9). The tracing mechanism is used when the modified code is run (step 270 of Figure 3 of the Drawings).

The apparatus recited in claim 40 includes "a storage device," such as the processor readable storage device disclosed in the Specification (p. 9, line 19). The claim further recites "one or more processors in communication with said storage device, said one or more processors perform a process." As discussed above, the apparatus can be, for example, a "computing devices… [with] one or more processors" (p. 9, line 18-19).

Claim 40 further recites "accessing a method" and "determining whether said method calls one or more different methods and can be called by a sufficient scope of one or more other methods." These features are similar to those recited in claim 1, as discussed above and are shown in at least Figure 5, step 286 and Specification p. 10, lines 10-21. Furthermore, the Specification discloses that the "method can be classified as complex if it satisfies two of the above criteria, or other similar criteria" (i.e. the method access level is public or package, the method calls another method, the method is non-synthetic) (p. 12, lines 17-23). Therefore, any criteria similar to these three criteria can be used to classify the method as complex for the purposes of modification.

Claim 40 further recites "tracing said method for a particular purpose if said method calls one or more different methods and can be called by a sufficient scope of one or more other methods." These features are similar to those recited in claim 13, as discussed above and shown in Figure 5, step 288 and Specification p. 15, line 6-9. If the method is determined to have satisfied these criteria, the method will be modified for tracing and the tracing will occur when the modified code is run.

The method recited in claim 47 includes "accessing a method; determining whether said method is complex, said step of determining includes determining that said method is complex if said method calls another method; and modifying said method for a particular purpose only if said method is determined to be complex." As discussed above, the Specification describes that a method may be determined to be complex if it satisfies the three criteria (i.e. the method access level is public or package, the method calls another method, the method is non-synthetic), or other similar criteria (p. 12, lines 17-23). If the method is determined to be complex, the method is modified as discussed above for claim 1 (see Figure 5, step 288).

Note that the explanations of the independent claims are not limited to those features referenced above by the Specification or Drawings. The features recited in the independent claims are disclosed throughout the Specification and Drawings, and the above descriptions merely serve as a concise

explanation of the subject matter defined in each of the independent claims.

CONCLUSION

As instructed on the Notification of Non-Compliant Appeal Brief, the defective section has been amended to provide a concise explanation of the subject matter defined in each independent claim by referring to the Specification and Drawings. Appellants respectfully request reconsideration of this section.

The Commissioner is authorized to charge any underpayment or credit any overpayment to Deposit Account No. 501826 for any matter in connection with this Amendment, including any fee for extension of time, which may be required.

Respectfully submitted,

Date: April 17, 2008

By: /Michelle Esteban/
Michelle Esteban
Reg. No. 59,880

VIERRA MAGEN MARCUS & DENIRO LLP
575 Market Street, Suite 2500
San Francisco, California 94105
Telephone: (415) 369-9660
Facsimile: (415) 369-9665